LSDT SOFTWARE FUNCTIONS LIBRARY REFERENCE GUIDE

LUMISYS

LUMISYS

LSDT Software Functions Library

Reference Guide

P/N 0066-022, Rev 10

March 9, 1999

Lumisys, Inc.
225 Humboldt Court
Sunnyvale, CA 94089

TABLE OF CONTENTS

TABLE OF CONTENTS **1**

1. INTRODUCTION **3**
1.1 Document Notation 3
1.2 Terminology 3
1.3 Related Documents 3
1.4 Environment Requirements 3
1.4.1 Minimum Hardware 3
1.4.1.1 DOS 3
1.4.1.2 Windows 3
1.4.2 Operating System 4
1.4.3 CPU Resources Used by the Data Control Board 4
1.4.4 LSDT Device Driver (DCB Interface Only) 4

2. LUMISCAN LSDT ARCHITECTURE **5**
2.1 Interfaces 5
2.1.1 DCB Interface 5
2.1.1.1 DCB Memory Mapping 5
2.1.2 SCSI Interface 6
2.2 Technical Features 6
2.2.1 Scan Mode and Pixel Resolution 6
2.2.1.1 Fixed 6
2.2.1.2 Variable 6
2.2.2 Pixel Depth 7
2.2.3 Pixel Format 7
2.2.4 Averaging Modes 7
2.2.5 Shifted Dynamic Range Mode 7
2.2.6 User LUT 7
2.2.7 6-Sheet Film Feeder 8
2.2.8 Bulk Autoloader Film Feeder 9
2.2.9 Barcode Reader 9
2.2.10 Film Present Sensor 9
2.3 Digitizer Capabilities 11

3. SOFTWARE FUNCTIONS **12**
3.1 General Setup Functions 12
3.2 Special Operation Functions 12
3.3 Scanning Functions 12
3.4 Post-Scan Functions 12
3.5 Miscellaneous Functions 13
3.6 Diagnostic Functions 13
3.7 Compatibility Functions 13

4. SOFTWARE APPLICATIONS **14**
4.1 Linking LSDT Software Functions 14
4.2 Example Code File 14
4.2.1 Finding Pixels Per Line 15

5. LIBRARY FUNCTIONS REFERENCE **16**
5.1 LS_ChangeTimer 16
5.2 LS_DiagnosticScan 16
5.3 LS_DriverStateString 17
5.4 LS_ErrorString 17
5.5 LS_FeedFilm 17
5.6 LS_GetAllTimers 18

5.7 LS_GetApiInfo 18
5.8 LS_GetDataCount 18
5.9 LS_GetDriverVector 18
5.10 LS_GetDrvInfo 19
5.11 LS_GetDrvVersion 19
5.12 LS_GetImageWindow 20
5.13 LS_GetPixelsPerLine 20
5.14 LS_GetStatus 20
5.15 LS_GetTimer 21
5.16 LS_GetVersion 21
5.17 LS_isBarCodeValid 21
5.18 LS_isFilmPresent 22
5.19 LS_LoadCalibrationTable 22
5.20 LS_LoadCorrectionLookUpTable 22
5.21 LS_LoadLookUpTable 22
5.22 LS_LoadLut 23
5.23 LS_LocateDriver 23
5.24 LS_MapImageData 24
5.25 LS_MapImageWindow 24
5.26 LS_ModelNameString 25
5.27 LS_Motor 25
5.28 LS_PixelDepthString 26
5.29 LS_PixelFormatString 26
5.30 LS_ReadBarCode 26
5.31 LS_ReadCalibrationTable 27
5.32 LS_ReadCorrectionLookUpTable 27
5.33 LS_ReadDiagnostic 27
5.34 LS_ReadImageIntoBuffer 28
5.35 LS_ReadImageIntoFile 28
5.36 LS_ReadImageIntoFile_Fast 29
5.37 LS_ReadImageToPointer 29
5.38 LS_ReadLookUpTable 30
5.39 LS_ReadLut 30
5.40 LS_RegisterCmd 31
5.41 LS_ResetDriverVariables 31
5.42 LS_ResetHardware 31
5.43 LS_ResetScan 32
5.44 LS_ScanModeString 32
5.45 LS_SetAveragingMode 32
5.46 LS_SetDevNum 32
5.47 LS_SetLaserMode 33
5.48 LS_SetNextScanParams 33
5.49 LS_SetPMTvoltage 34
5.50 LS_SetSDRMode 34
5.51 LS_SetTrim 34
5.52 LS_StartFilmScan 35
5.53 LS_StartFilmScanNoCLUT 35
5.54 LS_StartMotor 36
5.55 LS_StatusOfLastScanString 36
5.56 LS_StopMotor 36
5.57 LS_StopScanEject 37
5.58 LS_UnMapImageWindow 37
5.59 LS_WriteImageMemory 37

APPENDIX A: LUMISYS IMAGE HEADER FORMAT 38

APPENDIX B: FUNCTION ERROR CODES 39

1. INTRODUCTION

This document describes the functions in the LSDT Functions Software Library (LSDTFCNS.LIB). They provide a ***DOS*** high-level application interface to the LSDT digitizers. Refer to the LSDT API Dynamic Link Library Reference Guide (P/N 0068-105) for information on ***Windows 3.x / Windows 9x / Windows NT*** high-level application interface functions to the LSDT digitizers.

1.1 Document Notation

ALL_CAPS_IN_ITALICS – denotes a #DEFINE value. Almost all of the #DEFINEs you will need are in the following header files:

- LS_TYPES.H
  Data types and defines used by the library.
- LS_DT.H
  Defines, constants and structures used by the library, LSDT errors, scan modes, LSDT states (a.k.a. "phases"), and miscellaneous defines and structures.
- LS_FCNS.H
  Describes the LSDT functions and calling parameters.

If you don't find what you need in these two files, then look in one of the other include files in \INCLUDE.

1.2 Terminology

- DACQ Data Acquisition Board in the LSDT digitizer
- DCB ISA-based Data Control Board located in the PC
- DOS Any MS or PC DOS, 5.0 or later
- Host PC containing the DCB
- Win16 for Microsoft Windows 3.1, Windows For Workgroups 3.11, Windows 95, or Windows 98
- Win32 for Microsoft Windows 95, Windows 98, or Windows NT

1.3 Related Documents

LSDT Software Functions Library Reference Guide	P/N 0066-022
LSDT API Dynamic Link Library Reference Guide	P/N 0068-105
LUMISCAN 20 Operator's Reference Guide	P/N 0069-384
LUMISCAN 50 / 75 / 85 Operator's Reference Guide	P/N 0061-494
ACR-2000 Operator's Reference Guide	P/N 0070-711
LUMISCAN LSDT Film Digitizer / CR Reader Configuration Guide	P/N 0071-434

1.4 Environment Requirements

1.4.1 Minimum Hardware

1.4.1.1 DOS

- PC-386, 16 MHz or better
- 640KB RAM

1.4.1.2 Windows

- 486DX2/66 MHz (or faster, depending upon Windows Operating System)
- 8 MB RAM (or more, depending upon Windows Operating System)

1.4.2 Operating System

- MS-DOS 5.0 and above (using 16-bit driver w/ DCB)
- Windows 3.1 / Windows For Workgroups 3.11 (using 16-bit driver w/ DCB)
- Windows 95 / Windows 98 (using 16-bit driver w/ DCB)

1.4.3 CPU Resources Used by the Data Control Board

- Address Space
  - 32-Kbyte (0x8000 Hex) window
  - starting address 0xA0000-0xE8000 (switch selectable)
  - 0xD0000 – factory setting
- IRQ
  - 3, 4, 5, 6, 7 (jumper selectable)
  - 5 – factory setting
- I/O Address
  - 0x100-0x11F, 0x120-0x13F, 0x140-0x15F or 0x160-0x17F (switch selectable)
  - 0x100 – factory setting
  - **NOTE**: Older DCB models only supported addresses beginning at 0x100 and 0x120.
- DMA
  - None
- Bus Type
  - 8- or 16-bit ISA

1.4.4 LSDT Device Driver (DCB Interface Only)

The DOS LSDT device driver "LSDTVxxx.COM" must be installed on the host. At the command line or in AUTOEXEC.BAT, enter "LSDTVxxx" (where xxx specifies version number). This must be installed prior to starting Windows. For Windows 95 / Windows 98, the driver must be loaded by AUTOEXEC.BAT. The driver switch options are described in the **LUMISCAN** Operator's Reference Guides.

Digitizers with a SCSI interface do not use the LSDT device driver.

2. LUMISCAN LSDT ARCHITECTURE

The LumiScan LSDT family consists of film digitizers and CR phosphor plate readers using either a scanning laser beam and light collection cylinder or a CCD array with a proprietary illumination system. A pinch roller system is used to transport the medium being scanned.

NOTE: In this document, when we use the term "film", we mean a somewhat generic reference to "whatever is being scanned". Currently, several types of media are used in various digitizers and readers:

• Film (photographic negatives)	LS20 LS50, LS75, LS85	CCD Digitizer Laser Digitizers
• Phosphor-coated CR plates	ACR-2000	CR Reader
• Fluorescent CR Test patterns	ACR-2000	CR Reader

NOTE: Additionally, in this document, we use the term "digitizer", to mean a somewhat generic reference to either a (film) digitizer or a (CR) reader.

An LSDT digitizer houses a Data Acquisition Board (DACQ) which is connected to a Data Control Board (DCB) located in a host computer via a shielded 37-pin cable (for the SCSI interface, the DCB is located inside the digitizer). All operations are controlled by the DCB. Once a scan is initiated, the data acquisition is automatic, requiring no intervention. During a scan operation image data is written into a circular buffer in the image memory on the DCB. On digitizers with enough memory to hold a complete image it is possible to let the scan complete before transfer image data out of DCB image memory. However, if the image size is greater than the circular buffer size, then the situation is entirely different:

NOTE: For Images Greater Than the DCB Memory Size

The pinch roller media transport system does NOT allow the scan operation to stop before the end of the media. Therefore, the host application **MUST** read the image data out of the circular buffer as the data is acquired or the buffer will eventually overflow. When the DCB image memory circular buffer is full it wraps and begins overwriting the oldest data. If the host application has not read the old data when the image memory wraps, a "***DATA_LOST***" error is generated and data from the scan will truly be lost.

2.1 Interfaces

2.1.1 DCB Interface

In this interface, the DCB is located on an ISA card in a host computer. It connects to the digitizer via a shielded 37-pin cable. The card itself is a PC/AT ISA 8-Bit board requiring a 32K byte memory window (0xA0000-0xE8000) and 32 bytes of I/O space based at either 0x100, 0x120, 0x140 or 0x160 (older DCB boards only supported address at 0x100 and 0x120). An interrupt resource (IRQ) is also used and can be set to 3, 4, 5, 6 or 7. The 32K memory window is for access to the image memory and can be placed on any 32K boundary within the 0xA0000-0xE8000 address space. The DCB is populated with 4, 12 or 16 Megabytes of memory depending on the model of digitizer that will be connected.

The DCB is controlled by the LSDT device driver running on the host computer. The LSDT Software Functions Library (LSDTFCNS.LIB) provides a linkable DOS application interface to the LSDT digitizer (via the LSDT device driver). The library contains all the functions needed by a DOS application program to control the LSDT family of digitizers. Windows programs use the LSDT Dynamic Link Library (LSDTAPI.LIB) which provide substantially the same functions.

The library file "LSDTFCNS.LIB" contains the LSDT functions. An application program statically links with this library.

2.1.1.1 DCB Memory Mapping

The 32KB address space can be mapped into the application's address space. This allows direct access to the image data. The function LS_MapImageWindow is used to add the DCB memory to the applications address space. And LS_UnMapImageWindow is used to remove it. The function LS_GetImageWindow returns the physical address of the DCB memory, which is called by the LS_MapImageWindow function. The function

LS_MapImageData causes the driver to program the DCB hardware to place the image data specified by byte offset from the beginning of the image, into the DCB memory window. All mapping must be on 32K boundaries.

NOTE: For DOS 16-bit real mode programs, the "mapping" is actually the DCB physical address.

2.1.2 SCSI Interface

In this interface, the DCB is located inside the digitizer. It is part of a single board computer, also inside the digitizer, which in turn connects to a host computer via a standard SCSI cable.

NOTE: The LSDT Software Functions Library does not support this interface.

2.2 Technical Features

When you start a scan (LS_StartFilmScan), there are four things you must specify:

- Scan Resolution Mode — Fixed or Variable number of pixels per inch
- Pixel Resolution — Pixels Per Line (PPL) or Pixels Per Inch (PPI)
- Pixel Depth — 8 or 12 bits per pixel
- Pixel Format — LSB or MSB first

Additionally, there are other settable features such as averaging, shifted dynamic range, lookup tables, etc.

2.2.1 Scan Mode and Pixel Resolution

There are two scan modes: FIXED_RESOLUTION and VARIABLE_RESOLUTION. You specify which when you start a film scan. Note that all digitizers have minimum and maximum "pixels per line" and "pixels per inch" resolutions. These minimums and maximums vary according to the digitizer type, and in many cases, to the width of the film being scanned.

2.2.1.1 Fixed

FIXED_RESOLUTION means that the pixel resolution (or "pixels per inch") will be the same regardless of film width. The resolution will be determined by the "pixels per inch" argument passed to the LS_xxx functions. The number of pixels per line returned is determined by the film width with a maximum being imposed by the digitizer model.

When the requested pixels per inch times the film width causes the total number of pixels per line to exceed the digitizer model hardware limits, an error (***PIXELS_PER_LINE_OUT_OF_RANGE***) is returned and the film is reversed out of the digitizer. However, if the film width is just a little too large (< 1 inch), then the scan is performed by trimming the edges (instead of returning an error).

2.2.1.2 Variable

VARIABLE_RESOLUTION means that the pixel resolution (or "pixels per inch") will vary according to the width of the film. The resolution will be determined by the "pixels per line" (argument passed to the LS_xxx functions) over the entire width of the film. So regardless of film width, each scanned line will contain PPL pixels. For example, if the pixels per line is set to 1024 and the scan mode is variable, then a scanned line of film will contain 1024 pixels. If the width of film is 14", the pixel resolution will be 1024 ÷ 14 = 73 pixels/inch. If the film width is only 8 inches, a scanned line of film will still contain 1024 pixels; however, the pixel resolution will be higher: 1024 ÷ 8 = 128 pixels/inch.

NOTE: If the film is too narrow, a minimum pixel resolution may be encountered depending on the PPL requested for the scan. The effect of this limit is that the PPL scanned may be less than the PPL requested. The application should **always** check to ensure that the requested PPL are equal to the actual PPL. When should you check? Check right after data first starts becoming available from the scan. If fewer pixels per line are being delivered, the application should adjust accordingly.

In the case of the LS20 (since it is a CCD digitizer), if it scans a film which is too narrow it will adjust simply by scanning "air" on each side of the film.

2.2.2 Pixel Depth

The LSDT digitizer family can scan either 8-bit pixels or 12-bit pixels. In 8-bit mode, the scanning process is the same as 12-bit mode. The hardware digitizes raw 12-bit values and applies the corrections and calibrations to the 12-bit data. The 12-bit data is passed through the User LUT but only the low order 8 bits are placed into the DCB memory. To get good 8-bit data, the User LUT must map the raw 12-bit data down to the low 8 bits. See the section on the User LUT for details.

Some LSDT library functions require a "pixel depth" parameter. "pixel depth" can have one of two values: SCAN_8BITS or SCAN_12BITS. If “pixel depth” = SCAN_8BITS, then a scanned image results in 1 byte per pixel. If “pixel depth” = SCAN_12BITS then a scanned image results in 2 bytes per pixel. The data format for 12-bit pixels is with all the data in the lower bits.

NOTE: When scanning 8-bit pixels, a special User LUT must be loaded.

2.2.3 Pixel Format

There are two ways that 12-bit data is sent from a digitizer to the host. In both cases, the 12 bits of data are sent as a right-justified value in a 16-bit word. This 16-bit word is sent as Least Significant Byte (LSB) first, or Most Significant Byte (MSB) first. You specify which when you start a film scan.

2.2.4 Averaging Modes

There are three averaging modes: none, X only, and X+Y. For X averaging, two raw pixels are input and one pixel (the average) is output. For example, to scan with 1024 pixels per line with X averaging enabled, the hardware would scan 2048 raw pixels per line. Currently there is an absolute limit of 5120 raw pixels per line for all digitizer models. This limits all digitizers to a maximum of 2560 pixels per line with X averaging enabled. Not all digitizer models are capable of producing “absolute limit” pixels per line. Not all digitizer models support all averaging modes. For X+Y averaging, in addition to the X averaging described above, two raw lines are input and one line is output.

In general, scanning with averaging modes enabled improves the signal to noise ratio at the expense of a slower scanning speed.

NOTE: The CCD digitizers (LS20) use averaging to produce fewer pixels per inch. So for an LS20 to scan at 73 pixels per inch, X averaging is automatically enabled.

2.2.5 Shifted Dynamic Range Mode

The Shifted Dynamic Range (SDR) mode is an option that changes the mapping of pixel values to optical density. Normally pixel values are equal to the optical density scaled by 1000. So an optical density of 1.24 is represented by a pixel value of 1240.

i.e. Pixel = OD * 1000

This allows a 12-bit pixel value to span the optical density range 0.0 to 4.0.

When the SDR mode is enabled the pixel mapping is offset by 1000 counts:

i.e. Pixel = (OD * 1000) - 1000

This allows a 12-bit pixel value to span the optical density range 1.0 to 5.0.

Currently, only the LS85-SDR model supports the SDR mode option.

2.2.6 User LUT

While scanning, the final step before putting a pixel into the DCB memory is to pass it through the User LUT. The User LUT is a 4096-entry, 16-bit lookup table supplied by the “user” or host application. By default, the User LUT is loaded with a “linear” table where the first entry is 0, the second entry is 1, and so on until the last entry is loaded with 4095. The User LUT can be used for anything. Normally it is used to invert the pixels to change the meaning of black and white. By default the digitizer produces pixel values that correspond to optical density. Larger values mean darker. For display on a CRT it is often desirable to “invert” the pixel values, making a zero pixel black. This can be done by loading an “inverted” user LUT.

The default LUT on the DACQ is a 1:1 table, i.e. LUT[0] = 0, LUT[1] = 1, ... LUT[4095] = 4095. The user has the ability to load his own LUT to cause varying effects on the image. For example, to invert the image, the user would supply a LUT with "inverted" values, e.g. LUT[0] = 4095, LUT[1] = 4094, ... LUT[4095] = 0.

The user may scan with a "pixel depth" of 8 bits (SCAN_8BITS). If so, an 8-bit LUT should be downloaded to the DACQ. Each entry of the 8-bit LUT must be mapped into the LSB. A typical 8-bit LUT will be "step-like" in nature, e.g.

- LUT[0] .. LUT[15] = 0,
- LUT[16] .. LUT[31] = 1,
- ...
- LUT[4064] .. LUT[4079] = 254,
- LUT[4080] .. LUT[4095] = 255

NOTE: The previous 8-bit LUT "wastes" some of the available bits by mapping impossible Optical Densities into the 8 bits. A better method is to spread the pixel optical density across the 0 to 255 pixel values.

```
//  Get the Maximum Optical Density (maxOD) of the current digitizer

SHORT maxOpticalDensity;    // scaled by 1000, so OD=1.5 ==> 1500
DrvrInfoStruct DInfo;

    DInfo.infoType = DRVR_INFO_SYS_DATA;    // type of info to get
    DInfo.retBufferSize = sizeof(ScannerSystemDataStruct);
    if ((status = LS_GetDrvInfo(&DInfo)) == SUCCESS)
    { // ok, no error
        maxOpticalDensity = DInfo.infoBuffer.SystemInfo.Model.MaxOD;
    }

//  Create linear 8-bit User LUT mapping from 0 to maxOD into the
//  full 8 bits.  i.e. pixel OD=0 ==> 0, pixel OD= maxOD ==> 255
SHORT ix;
SHORT endOfVariablePartOfLUT;
SHORT LUT[ 4096];
SHORT capValue;
float stepValue;

    endOfVariablePartOfLUT = maxOpticalDensity;
    stepValue = 256.0 / (double) maxOpticalDensity;
    capValue = maxOpticalDensity / 16;  // convert 12-bit to 8-bit
    for (ix=0; ix <= endOfVariablePartOfLUT; ix++)
    { // create linear ramp from 0 to maxOD
        LUT[ix] = (SHORT) (stepValue * (float)ix);
    }
    for (ix=endOfVariablePartOfLUT; ix < 4096; ix++)
    { // fill in from maxOD to end of LUT
        LUT[ix] = capValue;
    }

//  now use LS_LoadLut() to load the "LUT" array into the digitizer
    status = LS_LoadLut(UNIVERSAL_USER_LUT_NUMBER, LUT);
```

2.2.7 6-Sheet Film Feeder

One of the film feeder options for the LSDT family of digitizers is an automatic 6-sheet feeder. This film feeder can load up to six sheets of film. The film feeder attaches to the front of the digitizer and provides six slots to hold film. The slots are numbered 1 through 6, counting from the back of the digitizer toward the front. Film is loaded in the order of the slot numbers. Slot 1 is special. Film placed in slot 1 will always be loaded into the digitizer. Slot 1 is the equivalent of the manual feed slot on printers.

The film feeder does NOT automatically load the film. The host application program is required to command the film feeder for each sheet of film. The LS_FeedFilm function must be called to activate the film feeder. When the LS_FeedFilm function is called, the next film slot is "opened". If the slot is empty, no film will be loaded.

The film feeder is reset manually by pressing a lever. The film feeder may be reset at any time. The host application can never tell the current position of the film feeder, or whether there is any film in the film feeder.

2.2.8 Bulk Autoloader Film Feeder

The other film feeder option for the LSDT family of digitizers is a bulk autoloader. This film feeder can load up to 100 sheets of film (fewer if they have barcodes attached). The film feeder attaches to the front of the digitizer and provides a single hopper in which to put film.

The film feeder does NOT automatically load the film. The host application program is required to command the film feeder for each sheet of film. The LS_FeedFilm function must be called to activate the film feeder. When the LS_FeedFilm function is called, the next film is picked up from the hopper and fed into the digitizer. If the hopper is empty, no film will be loaded. Unlike the 6-sheet feeder, the bulk autoloader requires that LS_FeedFilm be called for the first film as well as all the rest.

2.2.9 Barcode Reader

A barcode reader option is available on the LSDT family of digitizers. If present, the barcode reader is mounted to read barcodes as the film exits at the completion of a scan. Barcode data is valid from the time the barcode is read until the next scan is started (LS_StartFilmScan), however the barcode data can NOT be read until the digitizer has stopped scanning. The film eject does not have to complete first. A return value of HARDWARE_BUSY is returned if the digitizer is busy.

Example of reading the barcode data **after** all scan data has been read without error:

```
SHORT status;
CHAR barbuf[];
SHORT barLength;

barLength = sizeof(barbuf); // set length of buffer

//  loop until digitizer is NOT busy

while((status = LS_ReadBarCode(&barbuf, &barLng)) == HARDWARE_BUSY)
{
    barLength = sizeof(barbuf);     // reset length of buffer
    //  Note:   Every time LS_ReadBarCode is called barLength is changed
                kill some time, wait, do other things for ~1/4 second ...
}

if (status == 1)            // have VALID barcode data
{
    // barbuf[] has a 0 terminated string
    // barLng = strlen(barbuf)
}
else
if (status == 0)    // no barcode data was read
{
    //  so, nothing to do here
}
else
{
    //  "status" is an error code, use LS_ErrorString() to display
}
```

2.2.10 Film Present Sensor

A hardware film present sensor option is available on the LSDT family of digitizers. The film present sensor detects when film is loaded into the digitizer and is ready to be scanned. The sensor is independent of the film

feeder option. The sensor does NOT detect film IN the film feeder, ONLY film ready to scan. When used with the film feeder,

- First, the sensor is tested to see if an actual sheet of film was loaded.
- If not, then command the film feeder to load the next film and then test the sensor again to see if an actual sheet of film was loaded.

One possible use of the film present sensor is for the application program to poll the sensor while waiting for film. When film is detected, the application can automatically start the scanning process. Call LS_isFilmPresent to test the state of the sensor.

2.3 Digitizer Capabilities

Model	Type	Spot Size (μ)	Max Density (OD)	Averaging Modes Supported	Minimum Film Width (inch)	Maximum Film Width (inch)
LS20 (low res)	CCD film	350	3.2	XY	7	14
LS20 (hi res)	CCD film	175	3.2	XY	7	14
LS50	Laser film	210	3.6	None	8	14
LS75 (older)	Laser film	100	3.6	X, XY	8	14
LS75 (newer)	Laser film	100	3.6	X, XY	8	14
LS85	Laser film	50	4.1	X, XY	8	10
LS85-LF	Laser film	100	4.1	X, XY	11	14
LS85-SDR	Laser film	50	5.1	X, XY	8	10
ACR-2000	Laser CR	87	3.2	X, XY	6	14

Model	Type	Minimum Pixels Per Inch	Maximum Pixels Per Inch	Minimum Pixels per Line	Maximum Pixels per Line	Scan Speed (bytes / sec)	Data Rate (bytes / sec)
LS20 (low res)	CCD film	73	146	256	1024	40	331,128
LS20 (hi res)	CCD film	146	146	256	2048	81	331,128
LS50	Laser film	36	128	256	1140	115	235,520
LS75 (older)	Laser film	55	256	512	2048	115	471,040
LS75 (newer)	Laser film	55	300	512	4096	115	942,080
LS85	Laser film	36	730	256	5120	75	768,000
LS85-LF	Laser film	36	400	256	4096	75	614,400
LS85-SDR	Laser film	36	730	256	5120	75	768,000
ACR-2000	Laser CR	73	585	512	5120	50	409,600

3. SOFTWARE FUNCTIONS

This section organizes the API functions by type or usage. These categories are only general descriptions. Refer to the function descriptions for the details of their use.

3.1 General Setup Functions

Setup functions are typically used before scanning is started.

LS_isFilmPresent	Returns the status of the film present sensor.
LS_MapImageWindow	Maps the DCB memory window into program address space.
LS_ResetHardware	Resets the digitizer hardware and reinitializes the driver.
LS_ResetScan	Reinitializes the driver.
LS_SetDevNum	Selects one of multiple installed digitizers.

3.2 Special Operation Functions

Some of these functions change values where the defaults are normally used.

LS_GetApiInfo	Returns detailed version information of the library (LSDTFCNS.LIB).
LS_GetDrvInfo	Returns driver information, superset of LS_GetStatus.
LS_GetVersion	Returns the version number of the library (LSDTFCNS.LIB).
LS_LoadLut	Loads the specified hardware LUT.
LS_ReadLut	Reads the specified hardware LUT.
LS_SetNextScanParams	Sets special parameters for the next film scan.
LS_StopScanEject	Stops the current scan and ejects the film.

3.3 Scanning Functions

These routines are the main functions used to specify and obtain image data.

LS_FeedFilm	Tries to load the next film from film feeder hardware.
LS_MapImageData	Changes the offset of the image data mapped to the DCB window.
LS_ReadImageToPointer	Reads image data into application's memory.
LS_SetAveragingMode	Sets the averaging mode parameter for the next film scan.
LS_SetPMTvoltage	Sets the PMT bias voltage (changes the gain) for the next film scan.
LS_SetSDRMode	Sets the SDR mode for the next film scan.
LS_SetTrim	Sets how much should be trimmed (or added) to the film's width.
LS_StartFilmScan	Starts a film scan operation.

3.4 Post-Scan Functions

These functions are used after the scan operation completes.

LS_GetDataCount	Returns the number of bytes of image data scanned into DCB memory.
LS_GetPixelsPerLine	Returns pixels per line setting of the digitizer hardware.
LS_GetStatus	Returns general status information, subset of LS_GetDrvInfo.
LS_isBarCodeValid	Returns TRUE if barcode data is currently valid.
LS_ReadBarCode	Returns the data read by the optional barcode reader.
LS_UnMapImageWindow	Removes the DCB memory window from program address space.

3.5 Miscellaneous Functions

These functions are only used by some applications or while processing errors.

LS_DriverStateString	Returns ASCII string explanation for supplied driver state.
LS_ErrorString	Returns ASCII string explanation for supplied error code.
LS_ModelNameString	Returns ASCII string explanation for supplied model number.
LS_PixelDepthString	Returns ASCII string explanation for supplied pixel depth code.
LS_PixelFormatString	Returns ASCII string explanation for supplied pixel format code.
LS_ScanModeString	Returns ASCII string explanation for supplied scan mode.
LS_StatusOfLastScanString	Returns ASCII string explanation for supplied status of last scan.

3.6 Diagnostic Functions

These functions are almost **never** used by applications. Use of these functions without a detailed knowledge of the digitizer operation and hardware will more than likely produce strange results. Many of these functions produce results that are dependent on the exact digitizer model.

These functions are only included for completeness; they are NOT recommended.

LS_ChangeTimer	Sets specified timer with specified value.
LS_DiagnosticScan	Starts diagnostic scanning.
LS_GetAllTimers	Reads all hardware timer values.
LS_GetTimer	Reads the specified timer value.
LS_Motor	Runs (forward or backward) or stops the film transport motor.
LS_LocateDriver	Detects if DOS LSDT device driver is loaded or not.
LS_ReadDiagnostic	Diagnostic read from DCB image memory, including invalid data.
LS_RegisterCmd	Direct read or write to DCB hardware register.
LS_ResetDriverVariables	Resets driver variables to their detect cylinder width state.
LS_SetLaserMode	Sets Laser mode or LED illumination control.
LS_StartFilmScanNoCLUT	Special scan without using the hardware CLUT.
LS_WriteImageMemory	Diagnostic write to DCB image memory.

3.7 Compatibility Functions

These functions have all been superseded by simpler or newer functions. They still function as documented, but we recommend using only the functions listed above.

LS_GetImageWindow	Was used for DOS, not needed in Windows.
LS_LoadCalibrationTable	Use LS_LoadLut (UNIVERSAL_CAL_LUT_NUMBER, …).
LS_LoadCorrectionLookupTable	Use LS_LoadLut (UNIVERSAL_CORR_LUT_NUMBER, …).
LS_LoadLookUpTable	Use LS_LoadLut (UNIVERSAL_USER_LUT_NUMBER, …).
LS_ReadCalibrationTable	Use LS_ReadLut (UNIVERSAL_CAL_LUT_NUMBER, …).
LS_ReadCorrectionLookupTable	Use LS_ReadLut (UNIVERSAL_CORR_LUT_NUMBER, …).
LS_ReadImageIntoBuffer	Use LS_ReadImageToPointer.
LS_ReadImageIntoFile	Use LS_ReadImageToPointer, write into file yourself.
LS_ReadImageIntoFile_Fast	Use LS_ReadImageToPointer, write into file yourself.
LS_ReadLookUpTable	Use LS_ReadLut (UNIVERSAL_USER_LUT_NUMBER, …).
LS_StartMotor	Use LS_Motor (FORWARD_ or REVERSE_MOTER_DIRECTION).
LS_StopMotor	Use LS_Motor (STOP_MOTOR_DIRECTION).

4. SOFTWARE APPLICATIONS

4.1 Linking LSDT Software Functions

To use the LSDT software functions:

1) ***Include "LS_TYPES.H", "LS_DT.H", and "LS_FCNS.H" in your application*** (as in the code example below)**.**

2) Link with the "LSDTFCNS.LIB" file. An example Microsoft C6.0 make file follows:

```
INCL    =C:\INCLUDE             -- location of include files

all:    test.exe                -- your "main" executable

.c.obj:
    cl /AL /G2 /Gs  /W3 /Zp /c  /I$(INCL) $*.c

.obj.exe:
    link    test,,NUL.MAP,LSDTFCNS.LIB;

test.obj:   test.c

test.exe:   test.obj    LSDTFCNS.LIB
```

4.2 Example Code File

```
#include "LS_TYPES.H"       // standard LSDT data types
#include "LS_DT.H"          // standard LSDT defines
#include "LS_FCNS.H"        // LSDT functions

int status;

//  Scan film, if error, print it & exit
status = LS_StartFilmScan(1024, VARIABLE_MODE, SCAN_12BITS, LSB_FIRST);

//  check status
if (status != SUCCESS)
{
    printf("Could not start scan.  Error: %s\n", LS_ErrorString(status));
    exit(0);
}

//  Film is scanning!  Store in file, if error, print it & exit
status = LS_ReadImageIntoFile("IMAGE1.IMG");    // filename
if (status!= SUCCESS)
{
    printf("Could not scan film.  Error: %s\n", LS_ErrorString(status));
    exit(0);
}

printf("We scanned and stored an image to file!\n");
```

4.2.1 Finding Pixels Per Line

In many cases it is desirable to know the number of pixels per line for the current scan, before the scan is complete. When scanning in fixed resolution mode, or variable resolution mode with unknown film widths, the actual pixels per line is not known until the digitizer has detected the film size. So to find the actual pixels per line after the scan has started, check the driver status until the driver has found the film size and then call the LS_GetPixelsPerLine function. See the following:

```
SHORT GetActivePPL (VOID);      // function to determine when PPL is valid
SHORT status, pixels;

// Start the scan
if ((status = LS_StartFilmScan(1024, VARIABLE_MODE, SCAN_12BITS, LSB_FIRST))
    != SUCCESS) // error
{
    abort !
}

while((pixels = GetActivePPL()) == 0)
{
    // digitizer has NOT found the film size yet, so
    // kill some time OR go away and come back later
}

if (pixels < 0) // error - scan failed
{
    abort !
}

// have valid Pixels per line

// ---------------------------------------------------------------------
// GetActivePPL - Get the Pixels Per Line for the current scan
// ---------------------------------------------------------------------

SHORT GetActivePPL(VOID)
{
    LS_GetStatus(&driverStatus);
    if (driverStatus.driverState >= SCAN_PHASE_8)
    {
        return LS_GetPixelsPerLine();   // have PPL
    }
    else
        if (driverStatus.statusOfLastScan != SCAN_IN_PROGRESS)
            return FAILURE;                 // error

    return SUCCESS;                     // don't have PPL yet

}   // end of GetActivePPL
```

5. Library Functions Reference

The following is a list of ALL functions in the library in alphabetical order.

5.1 LS_ChangeTimer

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Changes one of the six timers on the DACQ.

```
int LS_ChangeTimer (int timerID, int value);
```

Input	Description	Values
timerID	Identifies one of six DACQ timers	***SCAN_MOTOR_CLOCK_TIMER*** ***FILM_MOTOR_CLOCK_TIMER*** ***PIXEL_CLOCK_TIMER*** ***DELAY_TO_FIRST_PIXEL_TIMER*** ***PIXELS_PER_LINE_TIMER*** ***LINES_PER_IMAGE_TIMER***
value	New timer value	***non-negative integer***

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
INVALID_PARAMETER	Invalid parameter
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.2 LS_DiagnosticScan

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Commands the driver to start scanning in diagnostic mode. Some scan modes are without film.

```
int LS_DiagnosticScan (int diagScanMode, int pixelDepth, int pixelFormat);
```

Input	Description	Values
diagScanMode	defines the type of diagnostic scan to perform	***SCAN_MODE_1*** ***SCAN_MODE_2*** ***SCAN_MODE_3*** ***SCAN_MODE_4*** ***SCAN_MODE_5*** ***SCAN_MODE_6*** ***SCAN_MODE_7*** ***DIAG_SCAN_AND_DISPLAY_CONTINUOUS***
pixelDepth	8 or 12 bits per pixel	***SCAN_8BITS*** ***SCAN_12BITS***
pixelFormat	MSB or LSB first	***MSB_FIRST*** ***LSB_FIRST (normal PC setting)***

Return Values	Description
SUCCESS	Function successful
PIXELS_PER_LINE_OUT_OF_RANGE	Invalid parameter
ILLEGAL_SCAN_MODE	Invalid parameter
ILLEGAL_BITS_PER_PIXEL	Invalid parameter
ILLEGAL_PIXEL_FORMAT	Invalid parameter

HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.3 LS_DriverStateString

Returns the driver state string associated with the input driverState code value.

```
char *LS_DriverStateString (int driverState);
```

Input	Description	Values
driverState	driver state code	value returned by LS_GetStatus – ***see LS_DT.H "Driver State DEFINES"***

Return Values	Description
NULL terminated character string	ASCII text description of driverState code

5.4 LS_ErrorString

Returns the error string associated with the input errorCode status. A user of this function may want to print out an ASCII error message for an error status received from some LSDT function.

```
char *LS_ErrorString (int errorCode);
```

Input	Description	Values
errorCode	ID code	***see LS_DT.H "Error DEFINES" & "Errors"***

Return Values	Description
NULL terminated character string	ASCII text description of error

5.5 LS_FeedFilm

Commands the film feeder hardware to load the next film. What this does depends upon what film loading equipment the digitizer has:

- 6-sheet film feeder hardware If it is armed (cocked) and film is present in the next slot, then the film will be dropped into position to be processed by the next scan operation.
- Bulk film loader If there is film in it, then the film will be picked up and dropped into position to be processed by the next scan operation.
- No loading equipmentNo effect.

```
int LS_FeedFilm (void);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.6 LS_GetAllTimers

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Returns the six DACQ timer values in a structure. If this function is called and the device driver has not been installed, all fields in the GetTimerStruct will be filled with zeros.

```
int LS_GetAllTimers (GetTimerStruct *timers);
```

Output	Description
timers	The structure is filled with the current values of the hardware timers. see LS_DT.H

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.7 LS_GetApiInfo

Returns a structure containing information regarding the library (LSDTFCNS.LIB).

```
VOID LS_GetApiInfo (VerInfoStruct far *ApiInfo);
```

Output	Description
ApiInfo	The structure is filled with version information. see LS_DT.H

5.8 LS_GetDataCount

Reads the Data Count register from the driver, which indicates the number of bytes of image data scanned into DCB memory. When the LSDT is scanning a film, the low order eight bits of the returned value are always zero.

NOTE: If the buffer has wrapped, the byte count returned will be larger than the amount of memory on the DCB.

```
long LS_GetDatacount (void);
```

Return Values	Description
positive number	Number of scanned image bytes
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.9 LS_GetDriverVector

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Returns the vector number of the currently installed device driver. The vector number of the device driver is internally maintained and used by the LSDT Software Functions Library.

```
int LS_GetDriverVector (void);
```

Return Values	Description
non-zero, positive integer (< 1024)	Installed driver interrupt vector
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.10 LS_GetDrvInfo

Returns dynamic and static status information from the driver. This is an expanded version of LS_GetStatus. Only up to retBufferSize bytes of data will be returned.

```
int LS_GetDrvInfo (DrvrInfoStruct far *drvrInfo)
```

Input	Description	Values
drvrInfo	Pointer to a DrvrInfoStruct	***- see LS_DT.H***
->infoType	What type of status to return	***DRVR_INFO_GET_STATUS*** ***DRVR_INFO_HW_SETTINGS*** ***DRVR_INFO_LAST_SCAN*** ***DRVR_INFO_GET_LAST_TIMERS*** ***DRVR_INFO_GET_SYS_DATA*** ***DRVR_INFO_PRIV_DATA*** ***DRVR_INFO_AGC_DATA*** ***DRVR_INFO_EDGE_SCAN_PARAMS***
->retBufferSize	How many bytes should be returned: DRVR_INFO_GET_STATUS DRVR_INFO_HW_SETTINGS DRVR_INFO_LAST_SCAN DRVR_INFO_GET_LAST_TIMERS DRVR_INFO_GET_SYS_DATA DRVR_INFO_PRIV_DATA DRVR_INFO_AGC_DATA DRVR_INFO_ EDGE_SCAN_PARAMS	 ***sizeof(InfoStatusStruct)*** ***sizeof(HWSettingsStruct)*** ***sizeof(LastScanStruct)*** ***sizeof(GetTimerStruct)*** ***sizeof(ScannerSystemDataStruct)*** ***sizeof(ModelPrivateDataStruct)*** ***sizeof(DrvrInfoAGCStruct)*** ***sizeof(EdgeScanParamsStruct)***

Output	Description
drvrInfo	Pointer to a DrvrInfoStruct
->bytesReturned	Actual number of bytes returned (should be equal to what was requested)
->infoBuffer	DrvrInfoStruct – a union of several status structures

Return Values	Description
SUCCESS	Function successfully returned status
NO_DRVR_HANDLE	LSDT Driver handle not available
INVALID_TYPE	Invalid Driver Status Type requested
INVALID_BUFFER_SIZE	Invalid Return Buffer Size

5.11 LS_GetDrvVersion

Returns the version ID of the currently loaded driver. A NULL terminated string is returned; allow 30 bytes.

```
int LS_GetDrvVersion (char *DrvVerString);
```

Output	Values	Description
DrvVerString	***ex: "3.52 Build 12"***	Pointer to ASCII string – Major (3), Minor (5), Maintenance (2), Build Number (12)

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.12 LS_GetImageWindow

This function is provided for compatibility with the Windows versions of the library that must map the physical address of the DCB Memory window to the user's space. In the DOS environment, this function will simply return a pointer to the DCB's 32KB memory window.

```
int LS_GetImageWindow (void far * far *mapAddr)
```

Output	Description
mapAddr	Will contain upon successful return a pointer to image memory.

Return Values	Description
SUCCESS	Function successfully mapped DCB to Image Memory
HARDWARE_FAILURE	Digitizer hardware malfunction
MEM_MAP_FAILURE	Memory Map failure

5.13 LS_GetPixelsPerLine

Returns the number of pixels per line of the most recently scanned image. Note: If you call this too early (before the driver has found the film edges and started scanning image data), the value returned is simply what you specified in the call to LS_StartFilmScan. It may very well NOT be what you are going to get in subsequent data. Wait until some image data is ready before calling this function.

```
int LS_GetPixelsPerLine (void);
```

Return Values	Description
positive number	Number of pixels per line or ***DEFAULT_PIXELS_PER_LINE*** (1024) if no image has been scanned
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.14 LS_GetStatus

Returns the current status of the LSDT from the device driver. The status will be returned in the structure pointed to by the function argument. This is exactly the same as LS_GetDrvInfo() where drvrInfo->infoType is set to DRVR_INFO_GET_STATUS

```
int LS_GetStatus (DriverStatusStruct *driverStatus);
```

Output	Description
driverStatus	See LS_DT.H – location to store returned status

Return Values	Description
SUCCESS	Function successfully returned status
INVALID_PARAMETER	Invalid parameter
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.15 LS_GetTimer

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Returns one of the six timers on the DACQ.

```
int LS_GetTimer (int timerID);
```

Input	Description	Values
timerID	Identifies one of six DACQ timers	***SCAN_MOTOR_CLOCK_TIMER*** ***FILM_MOTOR_CLOCK_TIMER*** ***PIXEL_CLOCK_TIMER*** ***DELAY_TO_FIRST_PIXEL_TIMER*** ***PIXELS_PER_LINE_TIMER*** ***LINES_PER_IMAGE_TIMER***

Return Values	Description
Positive integer	Timer value for specified timer
INVALID_PARAMETER	Invalid parameter
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.16 LS_GetVersion

Returns the character string indicating the version number of LSDTFCNS object library.

```
char * LS_GetVersion (void);
```

Return Values	Description
"LSDTFCNS Version x.xx"	x.xx reflects the current version number

5.17 LS_isBarCodeValid

Returns the status of the barcode data. A return value of 1 indicates that barcode data has become available since the last start of scan (LS_StartFilmScan). A return value of 0 indicates no barcode data is available or some kind of error (such as driver not loaded). This function makes use of the LS_GetDrvInfo function.

NOTE: Barcode data is valid from the time the barcode is read until the next scan is started; however, the barcode data can NOT be read until the digitizer has finished scanning data. If it is not available, the application program should wait until the digitizer is no longer in the "BUSY" state (i.e. while it is ejecting the film) and then try again (the barcode may be on the trailing edge of the film and not actually sensed until the eject phase).

```
int LS_isBarCodeValid (void);
```

Return Values	Description
1 (True)	A valid barcode has been read since the last start of scan
0 (***False***)	No barcode data (or error)

5.18 LS_isFilmPresent

Returns the status of the Film Present detector. A return value of 1 indicates the presence of film. A return value of 0 indicates no film, no film present detector, or some kind of error (such as driver not loaded).

```
int LS_isFilmPresent (void);
```

Return Values	Description
1 (True)	Detector sees film
0 (***False***)	No film (or error)

5.19 LS_LoadCalibrationTable

NOT RECOMMENDED – SUPERSEDED by LS_LoadLut

```
int LS_LoadCalibrationTable (int far *calibrationTable);
```

Input	Description	Values
calibrationTable	Location of Cal LUT values to load into the DACQ LUT	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.20 LS_LoadCorrectionLookUpTable

NOT RECOMMENDED – SUPERSEDED by LS_LoadLut

```
int LS_LoadCorrectionLookUpTable (int far *LUT);
```

Input	Description	Values
LUT	Location of CLUT values to load into the DACQ CLUT	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.21 LS_LoadLookUpTable

NOT RECOMMENDED – SUPERSEDED by LS_LoadLut

```
int LS_LoadLookUpTable (int far *LUT);
```

Input	Description	Values
LUT	Location of User LUT values to load into the DACQ ULUT	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.22 LS_LoadLut

Writes a user-supplied Lookup Table into the specified LUT on the DACQ. The caller needs to be sure to supply a large enough buffer. The data format of each entry is LSB first. This is a generalized version of older, specific LUT functions: LS_LoadCalibrationTable, LS_LoadCorrectionLookUpTable, and LS_LoadLookUpTable.

NOTE: This function can only be done when the digitizer is IDLE.

UNIVERSAL_CAL_LUT_NUMBER **NOT RECOMMENDED – DIAGNOSTIC USE ONLY**

LS_LoadLut loads a Calibration Lookup Table (Cal LUT) into the Data Acquisition Board (DACQ) on the LSDT. The number of entries in the table is equal to the maximum number of pixels per line, which for some digitizers may be as large as 5120 16-bit entries.

UNIVERSAL_CORR_LUT_NUMBER **NOT RECOMMENDED – DIAGNOSTIC USE ONLY**

LS_LoadLut loads a Correction Lookup Table (CLUT) into the Data Acquisition Board (DACQ) on the LSDT. The CLUT on the DACQ contains 4096 16-bit entries.

UNIVERSAL_USER_LUT_NUMBER

LS_LoadLut loads a User Lookup Table (ULUT) into the Data Acquisition Board (DACQ) on the LSDT. The User LUT on the DACQ contains 4096 16-bit entries, into which the DACQ indexes with the 12-bit (0..4095) optical density value for each pixel scanned.

```
int LS_LoadLut (unsigned int LutNumber, int far *LUT);
```

Input	Description	Values
LutNumber	LUT ID number	***UNIVERSAL_CAL_LUT_NUMBER*** ***UNIVERSAL_CORR_LUT_NUMBER*** ***UNIVERSAL_USER_LUT_NUMBER***
LUT	Pointer to LUT values to store into the DACQ	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.23 LS_LocateDriver

Returns SUCCESS if the interrupt handler for the DOS LSDT device driver is found. If the interrupt handler is not found, a status of DRIVER_NOT_INSTALLED is returned.

```
int LS_LocateDriver (void);
```

Return Values	Description
SUCCESS	Function successfully found interrupt handler
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.24 LS_MapImageData

Commands the driver to select the correct image memory page according to the imageOffset given. It is up to the application to read the image data from memory.

All the image data is accessed through a 32KB memory mapped window in the Memory Address Space of the ISA BUS. By factory default, the DCB is set up so that this window is at 0xD0000. It may be changed during installation. Data is always mapped on a 32KB boundary. To access a 10-MB image contained on this board, the application can read the data directly from the board but only 32KB at a time. After each 32KB chunk of image is read, the driver must be called to map to the next chunk of data. In the example file, SCANFILE, the use of these calls (LS_MapImageWindow and LS_MapImageData) is demonstrated. This usage is very similar to LS_ReadImageToPointer.

```
int LS_MapImageData (ReadStruct far *mapImageParam);
```

Input	Description	Values
mapImageParam	Pointer to Read Struct	***- see LS_DTR.H***
->imageOffset	Offset from beginning of image data to set image memory map page.	***- see LS_DT.H***
->readSize	Number of bytes application wishes to map. Should be 32,768 (one full memory page).	***- see LS_DT.H***
->buffer[1]	Unused for this function.	***- see LS_DT.H***

Output	Description
mapImageParam	Pointer to Read Struct
->bytesRead	Number of bytes that application can access. If less than one full memory page, then application should continue sending **LS_MapImageData** commands until either 1) Full memory page is available for access, 2) End Of Image status received, or 3) Error or Scan Aborted status received. Application can then directly access image memory data safely.

Return Values	Description
SUCCESS	Updated image data requested are available in Image Memory
HARDWARE_FAILURE	Digitizer hardware malfunction
NOT_ENOUGH_DATA	Number of bytes available is smaller than the number of bytes to read – the available bytes will be read and that number of bytes will be stored in ->bytesRead. More scan data is expected
END_OF_IMAGE	End of Image Reached; data is still available if ->bytesRead > 0
SCAN_ABORTED	Scan aborted
HARDWARE_BUSY	Not in right scan state for this command

5.25 LS_MapImageWindow

This function is provided for compatibility with Windows versions of the library that must map the physical address of the DCB Memory window to the user's program address space. In the DOS environment, this function will simply return a pointer to the DCB's 32KB memory window.

```
int LS_MapImageWindow (void far * far *imageWindowPtr);
```

Input	Description	Values
imageWindowPtr	Pointer to the 32KB DCB Memory window	***virtual memory address***

Return Values	Description
SUCCESS	Function successful
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.26 LS_ModelNameString

Returns the model name string associated with the input DACBModel model value. For example, an input value of 75 might be returned as "LS-75". You can obtain the model ID from LS_GetStatus or LS_GetDrvInfo.

```
char *LS_ModelNameString (int DACBModel);
```

Input	Description	Values
DACBModel	Model ID number	***e.g. 20, 50, 75, 76, 85, 86, 135***

Return Values	Description
NULL terminated character string	ASCII text description of digitizer model

```
DriverStruct    ds;

if (LS_GetStatus(&ds) == SUCCESS)
{
    printf("Digitizer Model: %s\n",
            LS_ModelNameString(ds.DACQModelRev.DACBModel));
}
```

5.27 LS_Motor

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Starts the film transport motor in the specified direction at the last motor speed or stops the motor. Use this generalized function rather than LS_StartMotor() and LS_StopMotor()

```
int LS_Motor (int motorCmd);
```

Input	Description	Values
motorCmd	Motor command code	***FORWARD_MOTOR_DIRECTION*** ***REVERSE_MOTOR_DIRECTION*** ***STOP_MOTOR_DIRECTION***

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
INVALID_PARAMETER	Invalid parameter
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.28 LS_PixelDepthString

Returns the pixel depth string associated with the input pixelDepth code value. You can obtain the current pixel depth from LS_GetStatus.

```
char *LS_PixelDepthString (int pixelDepth);
```

Input	Description	Values
pixelDepth	8-bit or 16-bit pixel specifier code	***SCAN_8BITS or SCAN_12BITS*** ***the pixelDepth value returned by LS_GetStatus – see LS_DT.H "Pixel Depth DEFINES"***

Return Values	Description
NULL terminated character string	ASCII string describing supplied pixel format code

5.29 LS_PixelFormatString

Returns the pixel format string associated with the input pixelFormat code value. You can obtain the current pixel format from LS_GetStatus.

```
char *LS_PixelFormatString (int pixelFormat);
```

Input	Description	Values
PixelFormat	Pixel format code	***MSB_FIRST or LSB_FIRST*** ***the pixelFormat value returned by LS_GetStatus – see LS_DT.H "Pixel Format DEFINES"***

Return Values	Description
NULL terminated character string	ASCII text description of ID code

5.30 LS_ReadBarCode

Returns the current barcode data, and a valid / not valid status. This function is called with a pointer to a buffer to receive the barcode data and a pointer to a length value. The length value defines the length of the user buffer on input, and on output the actual length of the returned barcode data. The maximum size of the barcode data is "LS_BARCODE_MAX_SIZE", not including the terminating NULL character ("0") which is appended to the return data if the user buffer is large enough. If the buffer is not large enough the terminating "0" will overwrite barcode data (i.e. the returned data always ends with a "0".) The return status is "1" for valid data, "0" with a return length of zero if the data is not valid; all other values are LSDT error codes.

NOTE: Barcode data is valid from the time the barcode is read until the next scan is started (LS_StartFilmScan), however the barcode data can NOT be read until the digitizer has stopped scanning. HARDWARE_BUSY is returned if the digitizer is busy. The data can be read during the film eject phase.

```
int LS_ReadBarCode (char *barString, int far *barStringSize);
```

Input	Description	Values
barStringSize	Address of dimensioned size of barString	***sizeof(barString);***

Output	Description	Values
barStringSize	Contains length of the barString not counting the final 0 terminator	***strlen(barString);***
barString	Location to store barcode data string	***0-terminated string***

Return Values	Description
0	Valid barcode data
1	Barcode data NOT valid
HARDWARE_BUSY	Digitizer is still scanning, try again later
other values	ERROR!

5.31 LS_ReadCalibrationTable

NOT RECOMMENDED – SUPERSEDED by LS_ReadLut

```
int LS_ReadCalibrationTable (int far *calibrationTable);
```

Input	Description	Values
calibrationTable	Location to store DACQ calibration table values	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.32 LS_ReadCorrectionLookUpTable

NOT RECOMMENDED – SUPERSEDED by LS_ReadLut

```
int LS_ReadCorrectionLookUpTable (int far *LUT);
```

Input	Description	Values
LUT	Location to store DACQ Correction LUT values	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.33 LS_ReadDiagnostic

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

This function is very similar to the LS_ReadImageToPointer function. The principle difference is that the LS_ReadDiagnostic function reads memory even if there is no image data present.

```
int LS_ReadDiagnostic (ReadStructPtr far *diagReadParamPtr);
```

Output	Description
diagReadParamPtr	Read Ptr Structure (see LS_DT.H)

Return Values	Description
SUCCESS	Function successfully read image
INVALID_PARAMETER	Invalid parameter

HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
HARDWARE_BUSY	This is returned when the digitizer is in a transitory state between the start of a scan and the actual scan. Try again!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.34 LS_ReadImageIntoBuffer

NOT RECOMMENDED – OBSOLETE

Commands the driver to read (copy) image data from the DCB memory into buffer memory on the host. This function is called once a scan starts. Typically, this function is called in a loop with readSize = 32K until an ***END_OF_IMAGE*** status is received.

This function uses an obsolete input parameter structure that defines parameter values in the same memory block as the returned scan data. Instead, you should use LS_ReadImageToPointer function with its "ReadStructPtr" struct.

```
int LS_ReadImageIntoBuffer (ReadStruct far *readParam);
```

Input	Description	Values
readParam	Read Structure (see LS_DT.H)	***a byte count (readSize) of 32,768 is recommended***

Return Values	Description
SUCCESS	Function successfully read image
INVALID_PARAMETER	Invalid parameter
NOT_ENOUGH_DATA	Number of bytes available is smaller than the number of bytes to read – the available bytes will be read and that number of bytes will be stored in ->bytesRead. More scan data is expected
END_OF_IMAGE	Same as ***NOT_ENOUGH_DATA*** except the digitizer has finished scanning the image
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed.
HARDWARE_BUSY	This is returned when the digitizer is in a transitory state between the start of a scan and the actual scan. Try again!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
DATA_LOST	Either the image is larger than DCB memory or a data overrun error has occurred.

5.35 LS_ReadImageIntoFile

NOT RECOMMENDED – OBSOLETE

Causes the driver to read (copy) an entire image from the DCB memory into a file on the host. As the data becomes available from the DCB, it is written into a host file. This file is in Lumisys image format (see Appendix A).

NOTE: This call will transfer the entire image off the DCB to a file. It calls LS_ReadImageToPointer().

```
int LS_ReadImageIntoFile (char far *fileName);
```

Input	Description	Values
fileName	path \ filename	***Any valid DOS path \ filename, e.g. XRAY1.IMG or C:\USERS\JACK\XRAY1.IMG***

Return Values	Description
SUCCESS	Function successfully read image into file
NOT_ENOUGH_DATA	Returned if no image in DCB memory
INVALID_PARAMETER	Invalid parameter
FILE_OPEN_ERROR	Could not open file

FILE_WRITE_ERROR	Could not write any or all bytes to file
HARDWARE_BUSY	LSDT not in ***IDLE*** state
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
DATA_LOST	Either the image is larger than DCB memory or a data overrun error has occurred

5.36 LS_ReadImageIntoFile_Fast

NOT RECOMMENDED – OBSOLETE

This function is the same as the LS_ReadImageIntoFile function, except implemented using the DCB mapping functions of the driver to read the incoming image directly from DCB memory.

```
int LS_ReadImageIntoFile_Fast (char far *fileName);
```

Input	Description	Values
fileName	path \ filename	***Any valid DOS path/filename, e.g. XRAY1.IMG or C:\USERS\JACK\XRAY1.IMG***

Return Values	Description
SUCCESS	Function successfully read image into file
NOT_ENOUGH_DATA	Returned if no image in DCB memory
INVALID_PARAMETER	Invalid parameter
FILE_OPEN_ERROR	Could not open file
FILE_WRITE_ERROR	Could not write any or all bytes to file
HARDWARE_BUSY	LSDT not in ***IDLE*** state
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
DATA_LOST	Either the image is larger than DCB memory or a data overrun error has occurred

5.37 LS_ReadImageToPointer

Commands the driver to read (copy) image data from the DCB memory into buffer memory on the host. This function may be called when a scan is occurring. An application will typically call this function in a loop with ->readSize = 32K until an ***END_OF_IMAGE*** status is received. This function differs from LS_ReadImageIntoBuffer in that a field of the ReadStructPtr structure contains a far pointer specifying where to store the image data – as opposed to the buffer starting in the ReadStruct structure itself.

```
int LS_ReadImageToPointer (ReadStructPtr far *readParam);
```

Input	Description	Values
readParam	Read Ptr Structure (see LS_DT.H)	***a byte count (readSize) of 32,768 is recommended***

Return Values	Description
SUCCESS	Function successfully read image
INVALID_PARAMETER	Invalid parameter
NOT_ENOUGH_DATA	Number of bytes available is smaller than the number of bytes to read – the available bytes will be read and that number of bytes will be stored in ->bytes_read. More scan data is expected

END_OF_IMAGE	Same as ***NOT_ENOUGH_DATA*** except the digitizer has finished scanning the image
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed.
HARDWARE_BUSY	This is returned when the digitizer is in a transitory state between the start of a scan and the actual scan. Try again!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed
DATA_LOST	Either the image is larger than DCB memory or a data overrun error has occurred

5.38 LS_ReadLookUpTable

NOT RECOMMENDED – SUPERSEDED by LS_ReadLut

```
int LS_ReadLookUpTable (int *LUT);
```

Output	Description	Values
LUT	Location to store DACQ ULUT values	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER NOT INSTALLED	Driver must be installed!

5.39 LS_ReadLut

Reads the specified LUT from the DACQ into a user-supplied buffer. The caller needs to be sure to supply a large enough buffer (up to 10,240 bytes). The data format of each entry is LSB first. This is a generalized version of older, specific and now obsolete LUT functions: LS_ReadCalibrationTable, LS_ReadCorrectionLookUpTable, and LS_ReadLookUpTable.

NOTE: This function can only be done when the digitizer is IDLE.

UNIVERSAL_CAL_LUT_NUMBER

LS_ReadLut reads the Calibration Lookup Table (Cal LUT) currently stored in the DACQ. Each entry in the table is an unsigned 16-bit integer. The number of entries in the table is equal to the number of pixels per line. The caller needs to be sure to supply a large enough buffer, 2 bytes for each pixel for the digitizer's maximum line size.

UNIVERSAL_CORR_LUT_NUMBER

LS_ReadLut reads the 4096 2-byte entries from the Correction Lookup Table (CLUT) currently stored in the DACQ.

UNIVERSAL_USER_LUT_NUMBER

LS_ReadLut reads the 4096 2-byte entries from the user Lookup Table (ULUT) currently stored in the DACQ. The User LUT on the DACQ contains 4096 16-bit entries, into which the DACQ indexes with the 12-bit (0..4095) optical density value for each pixel scanned.

```
int LS_ReadLut (unsigned int LutNumber, int far *LUT);
```

Input	Description	Values
LutNumber	LUT ID number	***UNIVERSAL_CAL_LUT_NUMBER*** ***UNIVERSAL_CORR_LUT_NUMBER*** ***UNIVERSAL_USER_LUT_NUMBER***

Output	Description	Values
LUT	Location to store DACQ LUT values	

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.40 LS_RegisterCmd

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Reads or writes directly to the digitizer hardware registers.

```
int LS_RegisterCmd (RegDataCmdStruct far *RegDataCmd);
```

Input / Output	Description
RegDataCmd	See LS_DT.H

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.41 LS_ResetDriverVariables

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Resets the digitizer driver scanning parameters to their initial settings used to detect cylinder width. This function is typically only used in diagnostics.

```
int LS_ResetDriverVariables (void);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.42 LS_ResetHardware

Resets the LSDT DACQ hardware and reinitializes the LSDT digitizer driver. This function is a superset of the LS_ResetScan function. The digitizer LUTs are reset to their initial state at the time the driver was loaded in addition to the LS_ResetScan operations. (See LS_ResetScan)

```
int LS_ResetHardware (void);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.43 LS_ResetScan

Reinitializes the LSDT digitizer driver. It does NOT reset the DACQ hardware (the DACQ retains its current Lookup and Calibration Tables). Stops the current scan if one is in progress. The LS_ResetScan functionality is automatically performed at the end of all normal scans, when a start scanning (LS_StartFilmScan) fails due to bad parameters, and when the edges of the film can not be found during a scan operation.

NOTE: The film is not ejected.

```
int LS_ResetScan (void);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.44 LS_ScanModeString

Returns the scan mode string associated with the input scanMode code value. You can obtain the scan mode from LS_GetStatus.

```
char *LS_ScanModeString (int scanMode);
```

Input	Description	Values
ScanMode	Scan mode value code	***scanMode value returned by LS_GetStatus – see LS_DT.H "Scan Mode DEFINES"***

Return Values	Description
NULL terminated character string	ASCII text description of supplied scanMode

5.45 LS_SetAveragingMode

Directs the digitizer driver to use the specified pixel averaging mode (see section 2.2.4) during the next scan only, regular or diagnostic.

```
int LS_SetAveragingMode (unsigned int averagingMode);
```

Input	Description	Values
AveragingMode	Specifies pixel averaging mode	***NO_AVERAGING*** ***X_ONLY_AVERAGING*** ***X_AND_Y_AVERAGING***

Return Values	Description
SUCCESS	Function successful
ILLEGAL_AVG_MODE	Illegal averaging mode

5.46 LS_SetDevNum

Allows you to select a particular DCB if you have more than one installed.

```
int LS_SetDevnum (unsigned int targetDevNum);
```

Input	Description	Values
targetDevNum	Which board	***1, 2, 3, …***

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.47 LS_SetLaserMode

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Directs the DACQ to set the scanning laser (or CCD illumination sub-system) to the specified mode.

```
int LS_SetLaserMode (unsigned int laserMode);
```

Input	Description	Values
laserMode	Specifies laser mode	***LASER_OFF***
		LASER_IDLE
		LASER_ON

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.48 LS_SetNextScanParams

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Sets special parameters for the next film scan or Mode 5 Diagnostic Scan. All values are reset to their default values after the next scan completes. This function ONLY affects the next scan.

```
int LS_SetNextScanParams (SetNextScanFilmStruct far *NextScanParams);
```

Input	Description	Values
NextScanParams	Pointer to a Parameter Block	
->ParamType	Parameter type	***NEXT_SCAN_TYPE_PMT_VOLT***
		NEXT_SCAN_TYPE_AUTOZERO
		NEXT_SCAN_TYPE_ENDSCAN
		NEXT_SCAN_TYPE_LASER
		NEXT_SCAN_TYPE_SDR
		NEXT_SCAN_TYPE_NOEDGE
		NEXT_SCAN_TYPE_AVG_MODE
->ParamValue	Value to set	

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.49 LS_SetPMTvoltage

Allows you to adjust the PMT high-voltage bias. This function ONLY affects the next scan. CR digitizers only.

```
int LS_SetPMTvoltage (unsigned char PMTadjust);
```

Input	Description	Values
PMTadjust	How much to adjust PMT voltage	***all values from 0 to 255, inclusive:*** ***0*** ***max Neg. voltage (less gain)*** ***128*** ***0V*** ***255*** ***max Pos. voltage (more gain)***

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.50 LS_SetSDRMode

Enables the Shifted Dynamic Range (SDR) mode for the next scan. After the next scan, the mode is automatically disabled. This is only used with an LS85 with the SDR option.

```
int LS_SetSDRMode (int SdrMode);
```

Input	Description	Values
SdrMode	Turn SDR mode ON or OFF	***SDR_MODE_ENABLE*** ***SDR_MODE_DISABLE***

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!

5.51 LS_SetTrim

Adds or subtracts pixels from where the film edges are detected. Its primary purpose is to account for the rounded corners on scanned media. You can also trim a bit extra to insure that no “air” is included on the image edges. This function ONLY affects the next scan.

```
int LS_SetTrim (int leftTrim, int rightTrim);
```

Input	Description
leftTrim	Number of mils (.001”) to trim from (if positive) or add to (if negative) left edge of film
rightTrim	Number of mils (.001”) to trim from (if positive) or add to (if negative) right edge of film

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.52 LS_StartFilmScan

Commands the driver to start scanning a film.

```
int LS_StartFilmScan (int pixelResolution,   int scanResMode
                      int pixelDepth,        int pixelFormat);
```

Input	Description	Values
pixelResolution	Pixels per line (scanMode = VARIABLE) Pixels per inch (scanMode = FIXED)	***256 ... 5120 -- depends on digitizer model*** ***36 ... 730 -- depends on digitizer model***
scanResMode	FIXED or VARIABLE mode	***FIXED_RESOLUTION*** ***VARIABLE_RESOLUTION***
pixelDepth	8 or 12 bits per pixel	***SCAN_8BITS*** ***SCAN_12BITS***
pixelFormat	MSB or LSB first	***MSB_FIRST*** ***LSB_FIRST (normal PC setting)***

Return Values	Description
SUCCESS	Function successful
PIXELS_PER_LINE_OUT_OF_RANGE	Invalid parameter
ILLEGAL_SCAN_MODE	Invalid parameter
ILLEGAL_BITS_PER_PIXEL	Invalid parameter
ILLEGAL_PIXEL_FORMAT	Invalid parameter
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.53 LS_StartFilmScanNoCLUT

NOT RECOMMENDED – DIAGNOSTIC USE ONLY

Commands the driver to start scanning a film. This function differs from the LS_StartFilmScan function because the pixel data bypasses the "CLUT" hardware. The CLUT corrects the pixel data value to the true optical density. LS_StartFilmScanNoCLUT is only used by diagnostic programs.

```
int LS_StartFilmScan (int pixelResolution,   int scanResMode
                      int pixelDepth,        int pixelFormat);
```

Input	Description	Values
pixelResolution	Pixels per line (scanMode = VARIABLE) Pixels per inch (scanMode = FIXED)	***256 ... 5120 -- depends on digitizer model*** ***36 ... 730 -- depends on digitizer model***
scanResMode	FIXED or VARIABLE mode	***FIXED_RESOLUTION*** ***VARIABLE_RESOLUTION***
pixelDepth	8 or 12 bits per pixel	***SCAN_8BITS*** ***SCAN_12BITS***
pixelFormat	MSB or LSB first	***MSB_FIRST*** ***LSB_FIRST (normal PC setting)***

Return Values	Description
SUCCESS	Function successful
PIXELS_PER_LINE_OUT_OF_RANGE	Invalid parameter
ILLEGAL_SCAN_MODE	Invalid parameter
ILLEGAL_BITS_PER_PIXEL	Invalid parameter
ILLEGAL_PIXEL_FORMAT	Invalid parameter

HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
SCAN_ABORTED_ERROR	Abort button pressed
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.54 LS_StartMotor

NOT RECOMMENDED – DIAGNOSTIC USE ONLY – SUPERSEDED by LS_Motor

Starts the film transport motor in the specified direction at the last specified motor speed.

```
int LS_StartMotor (int direction);
```

Input	Description	Values
direction	Direction to start motor	***FORWARD_MOTOR_DIRECTION*** ***REVERSE_MOTOR_DIRECTION***

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
INVALID_PARAMETER	Invalid parameter
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.55 LS_StatusOfLastScanString

Returns the status of last scan string associated with the input statusOfLastScan code value. You can obtain the status of last scan from LS_GetStatus.

```
char *LS_StatusOfLastScanString (int statusOfLastScan);
```

Input	Description	Values
statusOfLastScan	Status code value	***statusOfLastScan value returned by LS_GetStatus function – see LS_DT.H "Status of Last Scan DEFINES"***

Return Values	Description
NULL terminated character string	ASCII string describing supplied status code

5.56 LS_StopMotor

NOT RECOMMENDED – DIAGNOSTIC USE ONLY – SUPERSEDED by LS_Motor

Stops the film transport motor in the LSDT.

```
int LS_StopMotor (void);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_BUSY	LSDT not in ***IDLE*** state
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.57 LS_StopScanEject

Stops the current film scan and ejects the film.

```
int LS_StopScanEject (void);
```

Return Values	Description
SUCCESS	Function successful
HARDWARE_FAILURE	H/W failure – check power!
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.58 LS_UnMapImageWindow

This function is provided for compatibility with Windows versions of the library that must map the physical address of the DCB Memory window to the user's address space. In the DOS environment, this function simply returns SUCCESS.

```
int LS_UnMapImageWindow (void);
```

Return Values	Description
SUCCESS	Function successful
DRIVER_NOT_INSTALLED	The digitizer driver is not installed

5.59 LS_WriteImageMemory

NOT RECOMMENDED

Writes data into the DCB image memory. This function is used to perform memory tests. This function is ONLY available in the LSDT Software Functions Library.

```
int LS_WriteImageMemory (WriteParamStruct far *writeImageParamPtr);
```

Input	Description
WriteImageParamPtr	see LS_DT.H

Return Values	Description
SUCCESS	Function successful

Appendix A: LUMISYS IMAGE HEADER FORMAT

Byte Offset	Type	Description	Nr Bytes	Max Bytes	Example	Notes
0 – 803	N/A	Internal Use	804			* DO NOT USE! *
804 – 805	Integer	**Number of Images**	2		1	
806 – 807	Integer	**Pixels per Line**	2		2048	
808 – 809	Integer	**Lines per Image**	2		2486	
810 – 811	Integer	**Bits per Pixel**	2		12	
812 – 813	Integer	**Window**	2		4095	0 or 65535 imply not used
814 – 815	Integer	**Level**	2		2048	0 or 65535 imply not used
816 – 830	String	**Filename**	15	14	“LSDT.IMG”	
831 – 844	String	**Date**	14	13	“08/12/99”	
845 – 848	Long	**Time**	4		875029872	DOS Time Format (Seconds since 01/01/1970)
849 – 878	String	**Image Source**	30	29	“img_device"	Where image came from. Could be person, company, etc.
879 – 958	String	**Comment**	80	79	“Filtered & enhanced”	
959 – 989	String	**System Description**	31	30	“LS75”	
990 – 999	String	Header ID	10	N/A	“Lumisys”	Header format authentication (Exactly as shown!)
1000 – 1007	String	Version ID	8	N/A	“Hdr_Ver”	Version authentication (Exactly as shown!)
1008 – 1009	Integer	Version Number	2		4	Signifies version 4
1010 – 1011	Integer	**Byte Order**	2		0	0 = LSB, 1 = MSB, 2 = ?SB
1012 – 2047	N/A	Internal Use	1036			* DO NOT USE! *

These values are accessible in the include files LS_HDR.H and LS_HDROB.H. The latter is more object oriented and can be used with LS_HDROB.C. Support for images is supplied in LS_IMG.H and LS_IMG.C.

All header data is stored LSB first; the “Byte Order” field only pertains to the image data which follows.

“Max Bytes” means how many bytes you can put into this string field, not counting the terminating NULL character. Be sure to terminate all strings with a 0.

In general, for pre-existing headers, you should only change the bold-faced items above (“Number of Images” to “System Description”).

If “Hdr_Ver” is not found at 1000, then the Version Number at 1008 is undefined.

If creating a header from scratch, the Header ID, Version ID and Version Number are required, in addition to reasonable values of all the bold-faced items.

Version 3 Added Version ID and Version Number
Version 4 Added Byte Order

Appendix B: FUNCTION ERROR CODES

This section contains a complete list of possible decimal error codes returned by LSDT functions. The corresponding #defines are in LS_DT.H. If the meaning below is blank, then that error code is not used.

Code	*Meaning*	*Code*	*Meaning*	*Code*	*Meaning*
0	No error – successful	26	Illegal averaging mode	51	Map Window error
1	Illegal parameter buffer size	27	Illegal Bytes To Write	52	Window Not Mapped
2	Illegal driver function code	28	Data Lost (a)	53	Unmap Window error
3	Pixels per line out of range	29	SDR shift error	54	DLL Driver Rev Mismatch
4	Illegal scan mode	30	Multiple File Matches	55	Unknown Debug Subcode
5	Illegal bits per pixel	31	Bad File Pathname	56	Window Already Mapped
6	Illegal pixel format	32	Open File error	57	Illegal Image Offset
7	Illegal bytes to read	33	Read File error	58	SCSI General error
8	Hardware busy	34	Close File error	59	SCSI Device Not Present
9	Scan Timeout error	35	Illegal Parameter	60	SCSI LSDT Scanner Not Found
10		36		61	SCSI Inquiry Response Format Invalid
11	Not enough data	37	Cylinder Edge Detect Threshold Error	62	SCSI Sense error
12	End of image	38		63	SCSI Scanner Not Open
13		39	Film Edge Detect Bad Offset	64	Function Not Supported
14	Calibration table out of range	40	Film Edge Detect No Film	65	File Open error
15		41		66	File Write error
16		42	Film Edge Detect Bad Limits	67	File Close error
17		43	Special Functions Disabled (b)	68	File Read error
18		44	Film Edge Detect Bad Edges (Edge detect failed)	69	File Size error
19		45	Illegal Pixels Per Inch (c)	70	File Disk Space error
20		46	Fixed Resolution error (d)	71	File Not Specified
21	Hardware failure	47	Unknown error	72	End Of File
22	Scan aborted error	48	Memory Allocation error	73	Not Enough Disk Space
23	CLUT file error	49	Device Already Open	74	Invalid Filename
24	LUT file size error	50	Driver Not Installed	75	No Lumisys Image Header Found
25					

(a) Either the image is larger than the available memory on the DCB or in the “wrapping” mode the application has not taken the data fast enough to keep up with the incoming scan data. Pedal faster.

(b) Some functions can only be called in debug mode.

(c) For example, an LS-20 allows only 73 or 146 DPI

(d) You specified a value that will chop off or add more than 1”.